



Particle

MESH COMMUNICATION AND PARTICLE PRIMITIVES

**GETTING STARTED WITH THE WEB IDE**

**MESH PUBLISH & SUBSCRIBE**

**WORKING WITH PARTICLE PRIMITIVES**

**MANAGING DEVICES FROM THE CONSOLE**

**GETTING STARTED WITH THE WEB IDE**

**MESH PUBLISH & SUBSCRIBE**

**WORKING WITH PARTICLE PRIMITIVES**

**MANAGING DEVICES FROM THE CONSOLE**

# BUILD.PARTICLE.IO (WEB IDE)

Particle Apps

Current App  
**MEP-LS**  
*Optional description*

Files  
mep-ls.ino

**SHARE THIS REVISION**

**REMOVE APP**

My apps

Type to find

- Blink an LED
- Blink an LED1
- cloudTemp
- ledBox
- MEP-LS
- ParticleBlynkTest
- RaspberryBlinky
- runMem
- sdtest
- tftTester

**CREATE NEW APP**

Example apps

1. Blink an LED
2. Web-Connected LED
3. Function Variable
4. Publish
5. Subscribe
6. Tinker

mep-ls.ino

```
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22
23    delay(2000);
24
25    digitalWrite(GREEN, LOW);
26    digitalWrite(YELLOW, LOW);
27    digitalWrite(RED, LOW);
28 }
29
30 void loop() {
31 }
32
33 void respondToColor(const char *event, const char *data) {
34     receivedValue = data;
35
36     if (strcmp(data,"green") == 0) {
37         digitalWrite(GREEN, HIGH);
38         digitalWrite(YELLOW, LOW);
39         digitalWrite(RED, LOW);
40     } else if (strcmp(data,"yellow") == 0) {
41         digitalWrite(YELLOW, HIGH);
42         digitalWrite(GREEN, LOW);
43         digitalWrite(RED, LOW);
44     } else if (strcmp(data,"red") == 0) {
45         digitalWrite(RED, HIGH);
46         digitalWrite(GREEN, LOW);
47         digitalWrite(YELLOW, LOW);
48     }
49 }
```

Ready.

# BUILD.PARTICLE.IO (WEB IDE)

The screenshot shows the Particle Web IDE interface. On the left, there's a sidebar with various icons and sections like 'Particle Apps', 'My apps', 'Example apps', and a 'CREATE NEW APP' button. The main area has a dark background with a code editor window titled 'mep-ls.ino'. The code is as follows:

```
mep-ls.ino
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22
23    delay(2000);
24
25    digitalWrite(GREEN, LOW);
26    digitalWrite(YELLOW, LOW);
27    digitalWrite(RED, LOW);
28 }
29
30 void loop() {
31 }
32
33 void respondToColor(const char *event, const char *data) {
34     receivedValue = data;
35
36     if (strcmp(data, "green") == 0) {
37         digitalWrite(GREEN, HIGH);
38         digitalWrite(YELLOW, LOW);
39         digitalWrite(RED, LOW);
40     } else if (strcmp(data, "yellow") == 0) {
41         digitalWrite(YELLOW, HIGH);
42         digitalWrite(GREEN, LOW);
43         digitalWrite(RED, LOW);
44     } else if (strcmp(data, "red") == 0) {
45         digitalWrite(RED, HIGH);
46         digitalWrite(GREEN, LOW);
47         digitalWrite(YELLOW, LOW);
48     }
49 }
```

A blue modal box in the center-right says 'Info about your current app'.

# BUILD.PARTICLE.IO (WEB IDE)

The screenshot shows the Particle Web IDE interface. On the left, there's a sidebar with various icons and sections:

- Particle Apps**: Shows the current app "MEP-LS" with an optional description "Optional description". It includes buttons for "SHARE THIS REVISION" and "REMOVE APP".
- My apps**: A list of existing apps: Blink an LED, Blink an LED1, cloudTemp, ledBox, MEP-LS, ParticleBlynkTest, RaspberryBlinky, runMem, sdtest, tftTester. It also has a "CREATE NEW APP" button.
- Example apps**: A list of 6 examples: 1. Blink an LED, 2. Web-Connected LED, 3. Function Variable, 4. Publish, 5. Subscribe, 6. Tinker.

The main area displays the code for "mep-ls.ino":

```
mep-ls.ino
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22
23    delay(2000);
24
25    digitalWrite(GREEN, LOW);
26    digitalWrite(YELLOW, LOW);
27    digitalWrite(RED, LOW);
28 }
29
30 void loop() {
31 }
32
33 void respondToColor(const char *event, const char *data) {
34     receivedValue = data;
35
36     if (strcmp(data, "green") == 0) {
37         digitalWrite(GREEN, HIGH);
38         digitalWrite(YELLOW, LOW);
39         digitalWrite(RED, LOW);
40     } else if (strcmp(data, "yellow") == 0) {
41         digitalWrite(YELLOW, HIGH);
42         digitalWrite(GREEN, LOW);
43         digitalWrite(RED, LOW);
44     } else if (strcmp(data, "red") == 0) {
45         digitalWrite(RED, HIGH);
46         digitalWrite(GREEN, LOW);
47         digitalWrite(YELLOW, LOW);
48     }
49 }
```

A callout box on the right says "Apps you've created with the Web IDE". At the bottom, it says "Ready."

# BUILD.PARTICLE.IO (WEB IDE)

The screenshot shows the Particle Web IDE interface. On the left, there's a sidebar with icons for lightning bolt, checkmark, folder, and gear. Below these are sections for "Particle Apps", "My apps", "Example apps", and a "CREATE NEW APP" button. The "Particle Apps" section shows a current app "MEP-LS" with an optional description and buttons for "SHARE THIS REVISION" and "REMOVE APP". The "My apps" section lists various projects like "Blink an LED", "cloudTemp", and "tftTester". The "Example apps" section lists numbered examples from 1 to 6. The main area is a code editor titled "mep-ls.ino" containing the following Arduino-style pseudocode:

```
mep-ls.ino
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22
23    delay(2000);
24
25    digitalWrite(GREEN, LOW);
26    digitalWrite(YELLOW, LOW);
27    digitalWrite(RED, LOW);
28 }
29
30 void loop() {
31 }
32
33 void respondToColor(const char *event, const char *data) {
34     receivedValue = data;
35
36     if (strcmp(data, "green") == 0) {
37         digitalWrite(GREEN, HIGH);
38         digitalWrite(YELLOW, LOW);
39         digitalWrite(RED, LOW);
40     } else if (strcmp(data, "yellow") == 0) {
41         digitalWrite(YELLOW, HIGH);
42         digitalWrite(GREEN, LOW);
43         digitalWrite(RED, LOW);
44     } else if (strcmp(data, "red") == 0) {
45         digitalWrite(RED, HIGH);
46         digitalWrite(GREEN, LOW);
47         digitalWrite(YELLOW, LOW);
48     }
49 }
```

At the bottom of the code editor, it says "Ready."

A callout box on the right side of the screen contains the text: "Example Apps Particle provides".

# BUILD.PARTICLE.IO (WEB IDE)

The screenshot shows the Particle Web IDE interface. On the left, there's a sidebar with various icons and sections:

- Particle Apps**: Shows the current app "MEP-LS" with an optional description "Optional description". It includes buttons for "SHARE THIS REVISION" and "REMOVE APP".
- My apps**: A list of existing apps: Blink an LED, Blink an LED1, cloudTemp, ledBox, MEP-LS, ParticleBlynkTest, RaspberryBlinky, runMem, sdtest, tftTester. It also has a "CREATE NEW APP" button.
- Example apps**: A list of 6 examples: 1. Blink an LED, 2. Web-Connected LED, 3. Function Variable, 4. Publish, 5. Subscribe, 6. Tinker.

The main area displays the code for the "mep-ls.ino" sketch:

```
mep-ls.ino
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22
23    delay(2000);
24
25    digitalWrite(GREEN, LOW);
26    digitalWrite(YELLOW, LOW);
27    digitalWrite(RED, LOW);
28 }
29
30 void loop() {
31 }
32
33 void respondToColor(const char *event, const char *data) {
34     receivedValue = data;
35
36     if (strcmp(data,"green") == 0) {
37         digitalWrite(GREEN, HIGH);
38         digitalWrite(YELLOW, LOW);
39         digitalWrite(RED, LOW);
40     } else if (strcmp(data,"yellow") == 0) {
41         digitalWrite(YELLOW, HIGH);
42         digitalWrite(GREEN, LOW);
43         digitalWrite(RED, LOW);
44     } else if (strcmp(data,"red") == 0) {
45         digitalWrite(RED, HIGH);
46         digitalWrite(GREEN, LOW);
47         digitalWrite(YELLOW, LOW);
48     }
49 }
```

A large callout box on the right says "Code for your current app". At the bottom of the code editor, it says "Ready."

# BUILD.PARTICLE.IO (WEB IDE)

The screenshot shows the Particle Web IDE interface. On the left, there's a sidebar with icons for device management, file operations, and settings. The main area is divided into two sections: 'Particle Devices' on the left and a code editor on the right.

**Particle Devices:**

- Core**
  - carrot\_pants
- Photon**
  - avocado-toast
  - bacon\_mutant
  - brew\_buddy
  - brew\_buddy\_2
  - cyberspace\_button
  - library\_tester
  - office\_sensors
  - river-phoenix
  - rotator
  - tyrano\_janus
  - vampire\_wombat
  - wise-pirate
- Electron**
  - awesome-ninja
- Raspberry Pi**
  - raspberry
- #PartiBadge**
  - parti-alpaca
  - parti-badger
  - parti-bison
  - parti-egret
  - parti-lemur
  - parti-llama
  - parti-monkey
  - parti-nerfherder
  - parti-parrot
  - parti-penguin
  - parti-zebra
- #PartiBadge-Photon**
  - trash-panda

**mep-ls.ino**

```
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22    delay(2000);
23
24    digitalWrite(GREEN, LOW);
25    digitalWrite(YELLOW, LOW);
26    digitalWrite(RED, LOW);
27 }
28
29 void loop() {
30 }
31
32 void respondToColor(const char *event, const char *data) {
33     receivedValue = data;
34
35     if (strcmp(data,"green") == 0) {
36         digitalWrite(GREEN, HIGH);
37         digitalWrite(YELLOW, LOW);
38         digitalWrite(RED, LOW);
39     } else if (strcmp(data,"yellow") == 0) {
40         digitalWrite(YELLOW, HIGH);
41         digitalWrite(GREEN, LOW);
42         digitalWrite(RED, LOW);
43     } else if (strcmp(data,"red") == 0) {
44         digitalWrite(RED, HIGH);
45         digitalWrite(GREEN, LOW);
46         digitalWrite(YELLOW, LOW);
47     }
48 }
49 }
```

**Error:** No active devices. Please make at least one device active.

# BUILD.PARTICLE.IO (WEB IDE)

The screenshot shows the Particle Web IDE interface. On the left, there's a sidebar with icons for device management, file operations, and help. The main area is divided into two panes: a list of devices on the left and a code editor on the right.

**Particle Devices**

- C Core**
  - carrot\_pants (online)
- P Photon**
  - avocado-toast
  - bacon\_mutant
  - brew\_buddy
  - brew\_buddy\_2
  - cyberspace\_button
  - library\_tester
  - office\_sensors (online)
  - river-phoenix
  - rotator
  - tyrano\_janus
  - vampire\_wombat
  - wise-pirate
- E Electron**
  - awesome-ninja
- R Raspberry Pi**
  - raspberry
- #PartiBadge**
  - parti-alpaca
  - parti-badger
  - parti-bison
  - parti-egret
  - parti-lemur
  - parti-llama
  - parti-monkey
  - parti-nerfherder
  - parti-parrot
  - parti-penguin
  - parti-zebra
- P #PartiBadge-Photon**
  - trash-panda

**Code Editor (ep-ls.ino)**

```
ep-ls.ino
1 #define GREEN D0
2 #define RED D1
3 #define YELLOW D2
4
5 String receivedValue;
6
7 void setup() {
8     Particle.subscribe("output_color", respondToColor);
9     Particle.variable("receivedVal", receivedValue);
10
11    pinMode(GREEN, OUTPUT);
12    pinMode(YELLOW, OUTPUT);
13    pinMode(RED, OUTPUT);
14
15    digitalWrite(GREEN, HIGH);
16    delay(200);
17
18    digitalWrite(YELLOW, HIGH);
19    delay(200);
20
21    digitalWrite(RED, HIGH);
22    delay(2000);
23
24    digitalWrite(GREEN, LOW);
25    digitalWrite(YELLOW, LOW);
26    digitalWrite(RED, LOW);
27
28 }
29
30 void loop() {
31 }
32
33 void respondToColor(const char *event, const char *data) {
34     receivedValue = data;
35
36     if (strcmp(data,"green") == 0) {
37         digitalWrite(GREEN, HIGH);
38         digitalWrite(YELLOW, LOW);
39         digitalWrite(RED, LOW);
40     } else if (strcmp(data,"yellow") == 0) {
41         digitalWrite(YELLOW, HIGH);
42         digitalWrite(GREEN, LOW);
43         digitalWrite(RED, LOW);
44     } else if (strcmp(data,"red") == 0) {
45         digitalWrite(RED, HIGH);
46         digitalWrite(GREEN, LOW);
47         digitalWrite(YELLOW, LOW);
48     }
49 }
```

**Message Box**

All Your Devices  
(Blue dots are online devices)

Error: No active devices. Please make at least one device active.

# A WALKTHROUGH OF THE WEB IDE

DEMO

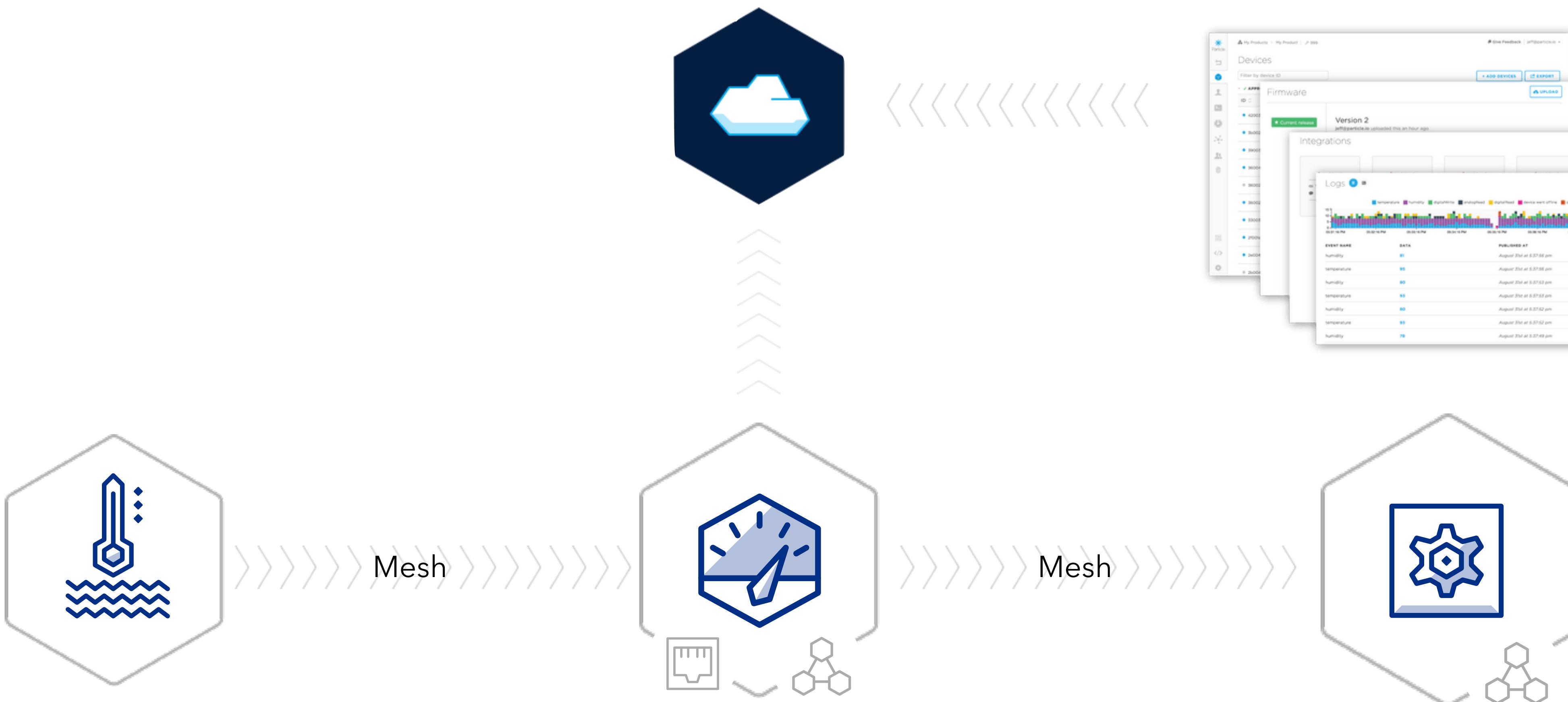
**GETTING STARTED WITH THE WEB IDE**

**MESH PUBLISH & SUBSCRIBE**

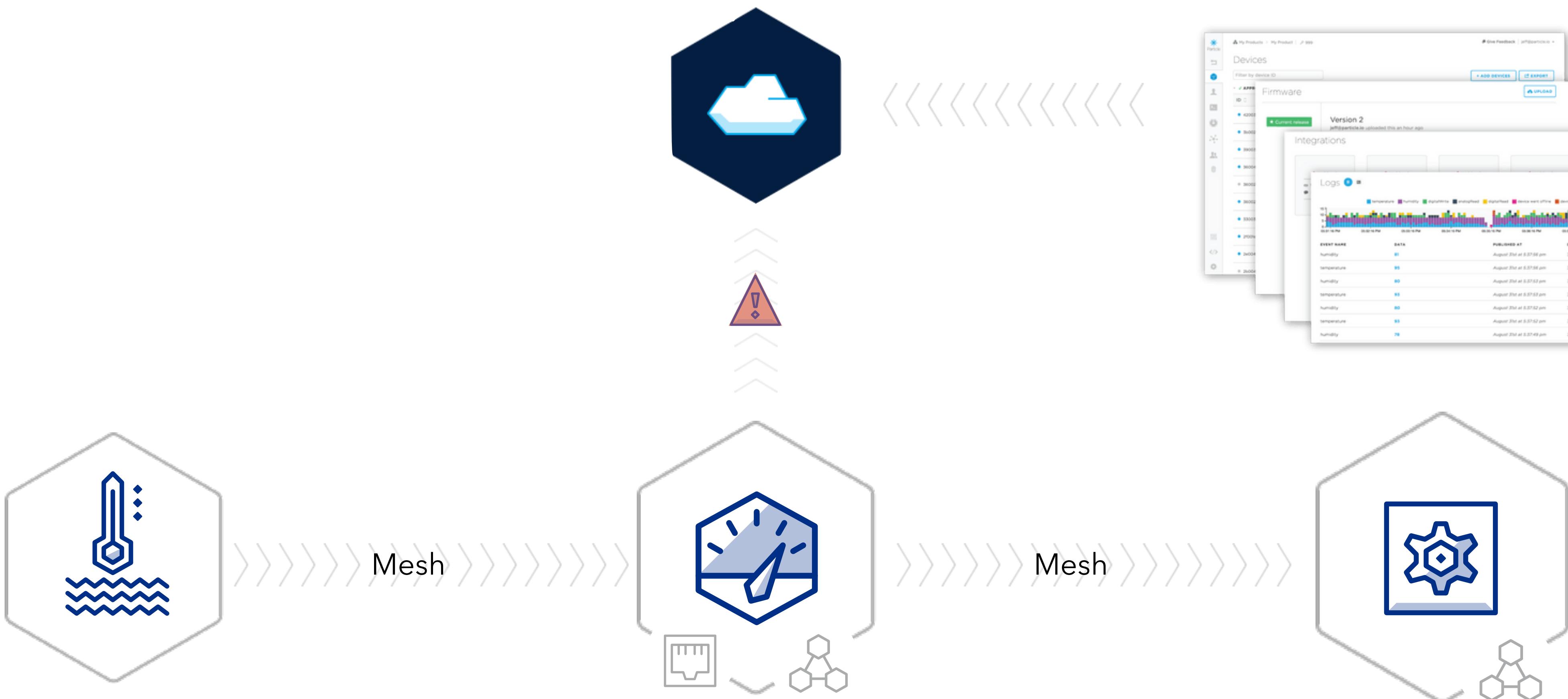
**WORKING WITH PARTICLE PRIMITIVES**

**MANAGING DEVICES FROM THE CONSOLE**

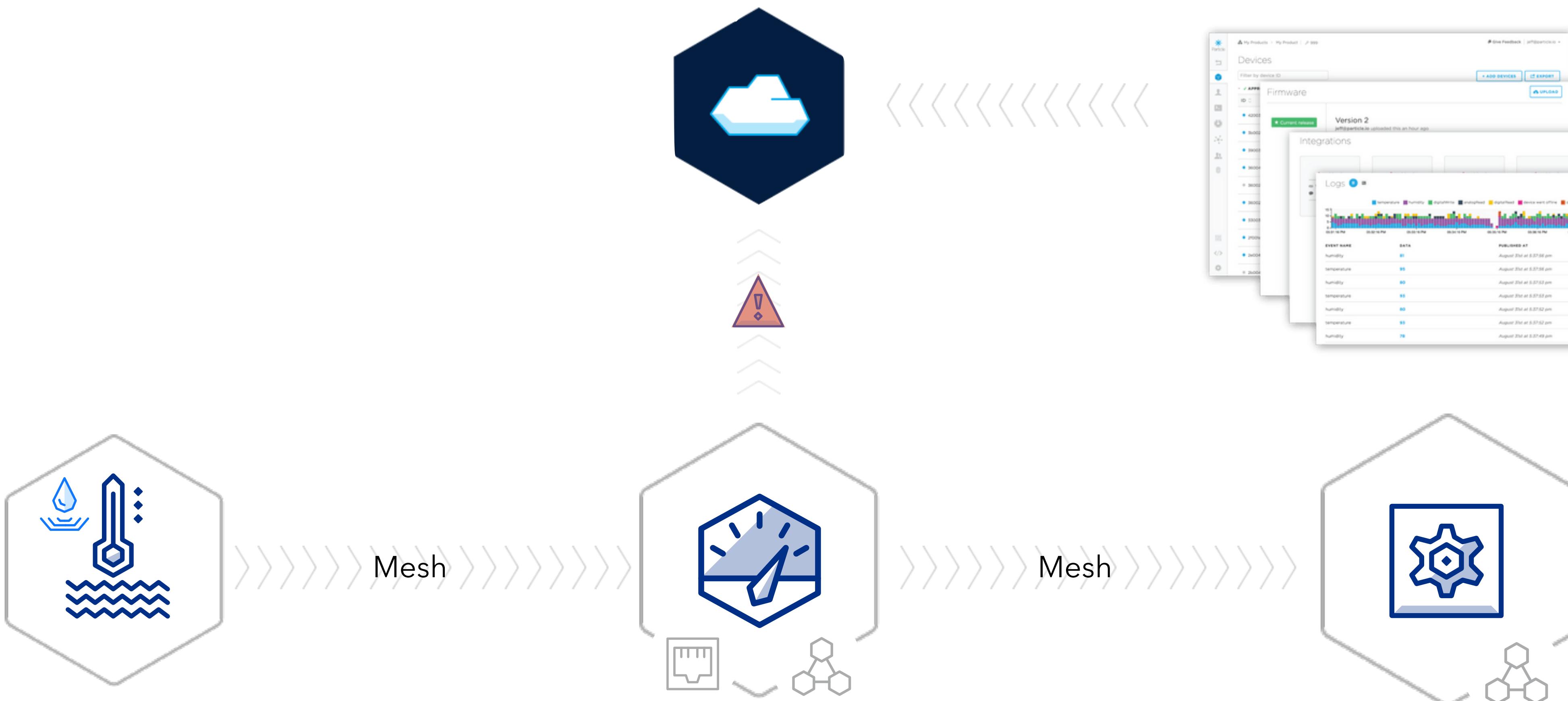
# PUB/SUB WITH PARTICLE MESH



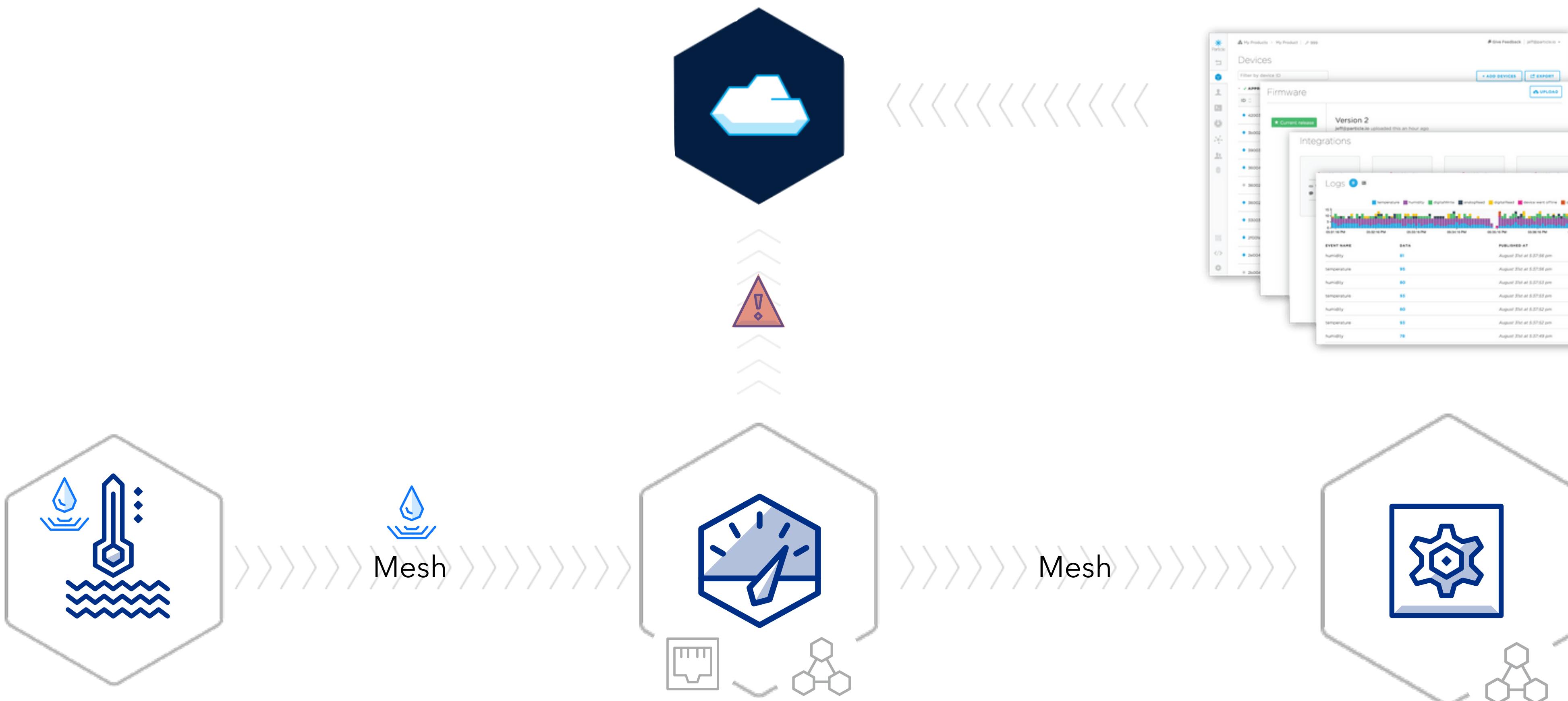
# PUB/SUB WITH PARTICLE MESH



# PUB/SUB WITH PARTICLE MESH



# PUB/SUB WITH PARTICLE MESH



# PUB/SUB WITH PARTICLE MESH



# PUB/SUB WITH PARTICLE MESH



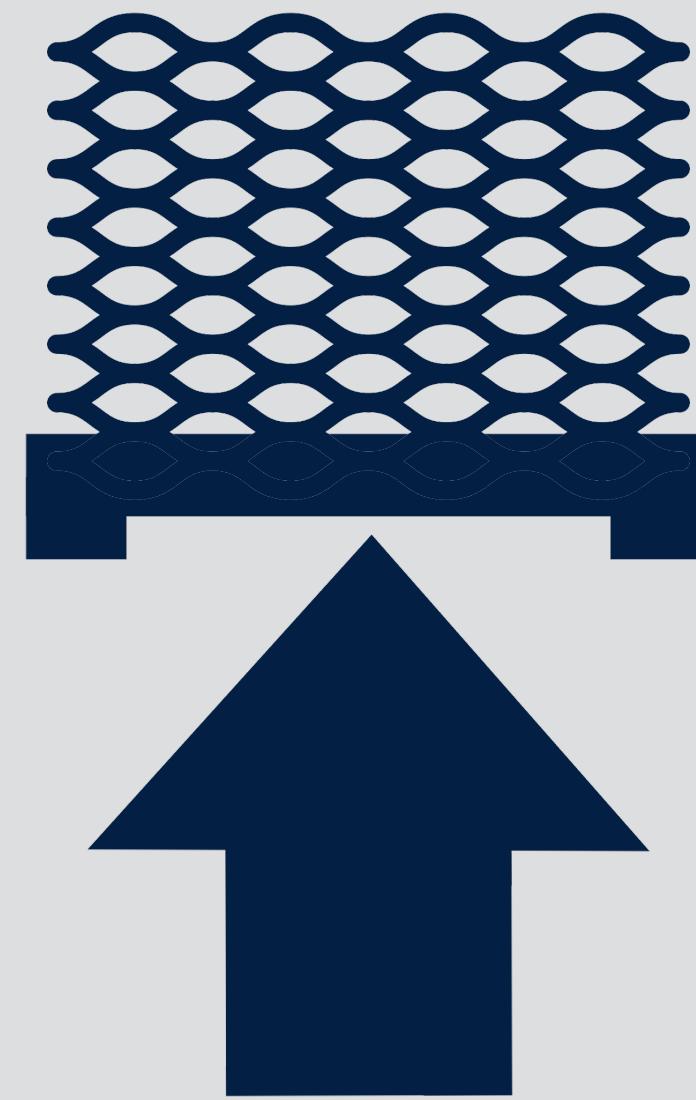
# PUB/SUB WITH PARTICLE MESH



# PARTICLE MESH FUNCTIONS

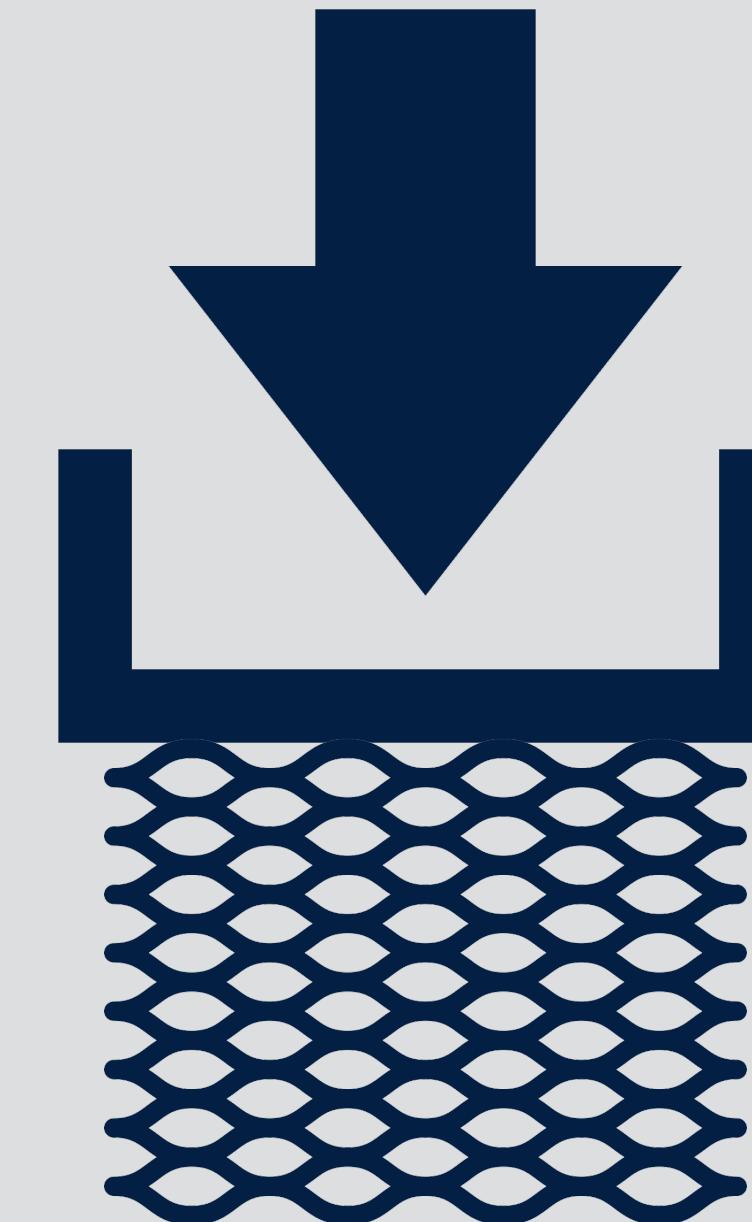
*Broadcast an event to all devices in a  
Mesh network*

**Mesh.publish()**



*Listen for events published to the Mesh  
network*

**Mesh.subscribe()**



# MESH.PUBLISH()

## What it does:

**Publish an event that will be forwarded to all registered listeners on the local Particle mesh network.**

## Why it's cool:

- \*Enables mesh network communication
- \*Works even when the network isn't connected to the cloud

## Usage notes:

- \*63 characters max for event names

```
double tempC = 0;

void setup()
{
    Particle.variable("temp", tempC);

    pinMode(A0, INPUT);
}

void loop()
{
    analogvalue = analogRead(A0);
    tempC = (((analogvalue * 3.3) / 4095) - 0.5) * 100;

    if (tempC > 120)
    {
        Mesh.publish("temp/critical", tempC);
    }
    else if (tempC > 80)
    {
        Mesh.publish("temp/warning", tempC);
    }
}
```

# MESH.SUBSCRIBE()

## What it does:

**Subscribe to events published by devices  
on the local mesh network.**

## Why it's cool:

- \*Enables mesh network communication
- \*Works even when the network isn't connected to the cloud

## Usage notes:

- \*Subscriptions work like prefix filters, meaning you can capture multiple publish events via clever naming.

```
void setup()
{
    // Subscribes to temp/warning AND temp/critical
    Mesh.subscribe("temp", handleTemp);
}

void handleTemp(const char *event, const char *data)
{
    double temp = extractTemp(data);

    if (temp > 120)
    {
        deactivatePump();
    }
    else if (temp > 80)
    {
        reducePumpSpeed();
    }
}
```

**GETTING STARTED WITH THE WEB IDE**

**MESH PUBLISH & SUBSCRIBE**

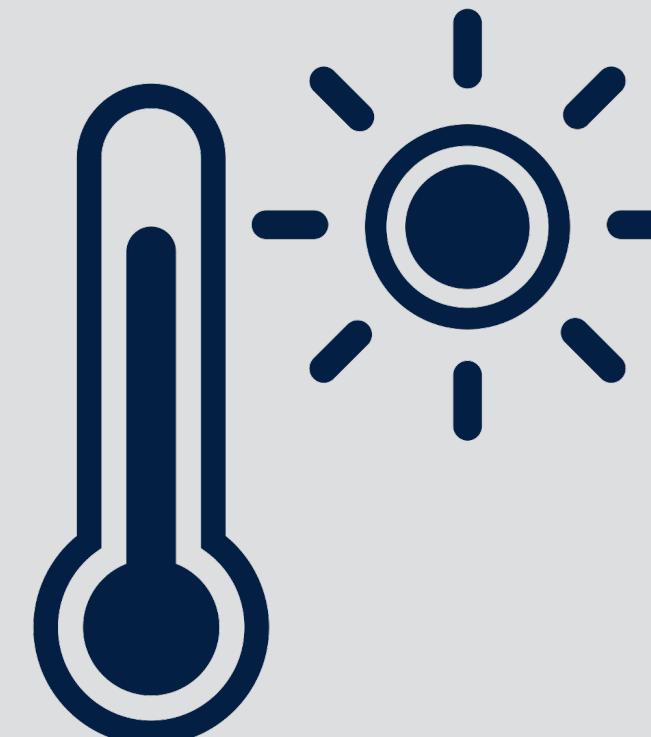
**WORKING WITH PARTICLE PRIMITIVES**

**MANAGING DEVICES FROM THE CONSOLE**

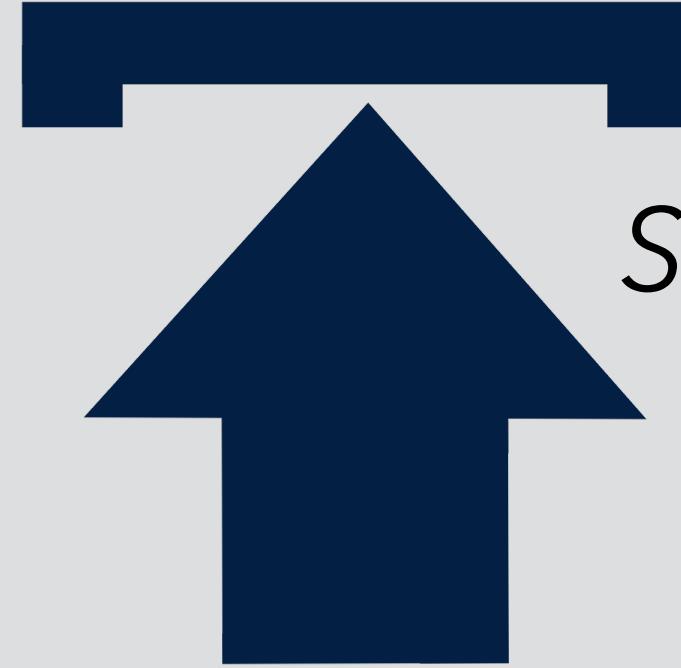
# PARTICLE CLOUD FUNCTIONS



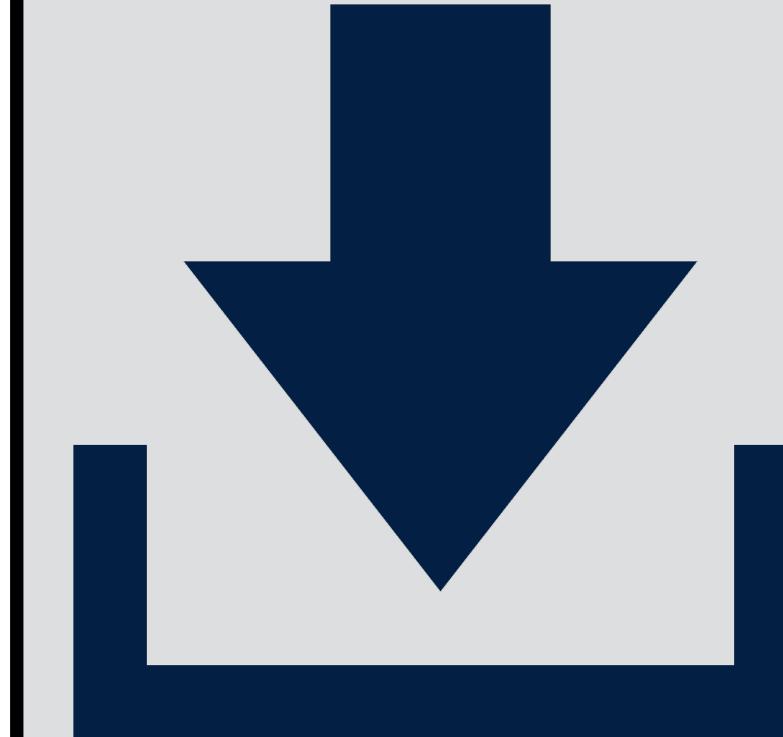
*Call a function, remotely*  
**Particle.function()**



*Fetch a variable, remotely*  
**Particle.variable()**



*Send an event to the cloud*  
**Particle.publish()**



*Listen for events*  
**Particle.subscribe()**

# PARTICLE.VARIABLE()

## What it does:

**Expose a firmware variable to the cloud**

## Why it's cool:

- \* Can be fetched via the Device Cloud API
- \* Viewable from the Device Console

## Usage notes:

- \* 20 variables max.
- \* 12 character limit per variable name

```
int analogvalue = 0;
double tempC = 0;

void setup()
{
    // variable name max length is 12 characters long
    Particle.variable("analogvalue", analogvalue);
    Particle.variable("temp", tempC);

    // Setup for Sensor on A0
    pinMode(A0, INPUT);
}

void loop()
{
    // Read the analog value of the sensor
    analogvalue = analogRead(A0);

    // Convert the reading into degrees Celsius
    tempC = (((analogvalue * 3.3)/4095) - 0.5) * 100;
    delay(200);
}
```

# PARTICLE.VARIABLE()

## What it does:

**Expose a firmware variable to the cloud**

## Why it's cool:

- \* Can be fetched via the Device Cloud API
- \* Viewable from the Device Console

## Usage notes:

- \* 20 variables max.
- \* 12 character limit per variable name

```
int analogvalue = 0;  
double tempC = 0;  
  
void setup()  
{
```

```
# EXAMPLE REQUEST IN TERMINAL  
# Device ID is 0123456789abcdef  
# Your access token is 123412341234  
curl "https://api.particle.io/v1/devices/0123456789abcdef/  
analogvalue?access_token=123412341234"  
curl "https://api.particle.io/v1/devices/0123456789abcdef/  
temp?access_token=123412341234"  
  
# In return you'll get something like this:  
960  
27.44322344322344
```

```
//Convert the reading into degrees Celsius  
tempC = (((analogvalue * 3.3)/4095) - 0.5) * 100;  
delay(200);  
}
```

# PARTICLE.FUNCTION()

## What it does:

**Expose a firmware function to the cloud**

## Why it's cool:

- \* Can be called via the Device Cloud API
- \* Callable from the Device Console

## Usage notes:

- \* 15 functions max.
- \* 12 character limit per function name

```
int togglePump(String command);

void setup()
{
    // register the cloud function
    Particle.function("togglePump", togglePump);
}

// this function automagically gets called upon a matching
POST request
int togglePump(String command)
{
    if (command == "on")
    {
        activateWaterPump();
    }
    else
    {
        deactivatePump();
    }

    return 1;
}
```

# PARTICLE.FUNCTION()

## What it does:

**Expose a firmware function to the cloud**

## Why it's cool:

- \* Can be called via the Device Cloud API
- \* Callable from the Device Console

## Usage notes:

- \* 15 functions max.
- \* 12 character limit per function name

```
int togglePump(String command);

void setup()
{
    # API Call
    # GET /v1/devices/{DEVICE_ID}/{VARIABLE}

    # EXAMPLE REQUEST IN TERMINAL
    # Device ID is 0123456789abcdef
    # Your access token is 123412341234
    curl "https://api.particle.io/v1/devices/0123456789abcdef/
analogvalue?access_token=123412341234"
    curl "https://api.particle.io/v1/devices/0123456789abcdef/
temp?access_token=123412341234"

    # In return you'll get something like this:
    960
    27.4432234432234

    }

    return 1;
}
```

# PARTICLE.PUBLISH()

## What it does:

**Publish an event that will be forwarded to all registered listeners.**

## Why it's cool:

- \*Enables device-to-device communication
- \*Viewable from the Device Console

## Usage notes:

- \*63 characters max for event names
- \*Events are public by default, but can be marked as private.

```
double tempC = 0;

void setup()
{
    Particle.variable("temp", tempC);

    pinMode(A0, INPUT);
}

void loop()
{
    analogvalue = analogRead(A0);
    tempC = (((analogvalue * 3.3) / 4095) - 0.5) * 100;

    if (tempC > 120)
    {
        Particle.publish("temp/critical", tempC);
    }
    else if (tempC > 80)
    {
        Particle.publish("temp/warning", tempC);
    }
}
```

# PARTICLE.PUBLISH()

## What it does:

**Publish an event that will be forwarded to all registered listeners.**

## Why it's cool:

- \* Enables device-to-device communication
- \* Viewable from the Device Console

## Usage notes:

- \* 63 characters max for event names
- \* Events are public by default, but can be marked as private.

```
double tempC = 0;

void setup()
{
    # API Call
    # GET /v1/events/{EVENT_NAME}

    # EXAMPLE REQUEST
    curl -H "Authorization: Bearer {ACCESS_TOKEN_Goes_Here}" \
        https://api.particle.io/v1/events/temp/critical

    # Will return a stream that echoes text when your event is
    # published
    event: temp/critical
    data:
    {
        "data": "125",
        "ttl": "60",
        "published_at": "2018-05-28T19:20:34.638Z",
        "deviceid": "0123456789abcdef"
    }

    Particle.publish("temp/warning", tempC);
}
```

# PARTICLE.SUBSCRIBE()

## What it does:

**Subscribe to events published by devices.**

## Why it's cool:

- \*Enables device-to-device communication
- \*Non-IoT devices can also trigger events

## Usage notes:

- \*4 subscribe handlers per device, max
- \*Subscriptions work like prefix filters, meaning you can capture multiple publish events via clever naming.

```
void setup()
{
    // Subscribes to temp/warning AND temp/critical
    Particle.subscribe("temp", handleTemp);
}

void handleTemp(const char *event, const char *data)
{
    double temp = extractTemp(data);

    if (temp > 120)
    {
        deactivatePump();
    }
    else if (temp > 80)
    {
        reducePumpSpeed();
    }
}
```

# LOCAL MESH PUB/SUB VS. PARTICLE CLOUD PUB/SUB

## **Mesh Pub/Sub is for local messages**

### **Use Mesh Pub/Sub When:**

- \* You need to communicate between devices *only* on a mesh
- \* You need messages to be sent as fast as possible
- \* You need to communicate between devices when a connection to the cloud is unavailable.
- \* It's ok that not every message is delivered.

## **Particle Pub/Sub is for everything else**

### **Use Particle Pub/Sub When:**

- \* You need to communicate between mesh networks or with devices not on a mesh network
- \* You're publishing events to webhooks or cloud integrations (Azure, Google Cloud, etc.)
- \* You need some QOS in message delivery (retry attempts, etc.)

**GETTING STARTED WITH THE WEB IDE**

**MESH PUBLISH & SUBSCRIBE**

**WORKING WITH PARTICLE PRIMITIVES**

**MANAGING DEVICES FROM THE CONSOLE**

Particle

Devices

Number of devices: 27

ID	Type	Name	Last Handshake	...
3b001e000247363339343638	(P) Photon	trash-panda	7/19/18 at 8:57am	...
190028000247343138333038	(P) Photon	office_sensors	7/19/18 at 8:24am	...
22001f00147353136383631	(P) Photon	rotator	7/18/18 at 8:29pm	...
36003b000547363339343638	(P) Photon	wise-pirate	7/18/18 at 4:45pm	...
54ff6c066672524814341167	(C) Core	carrot_pants	7/17/18 at 5:19pm	...
220044000347363339343638	(P) Photon	avocado-toast	7/17/18 at 9:20am	...
1a0038001647373334363431	(E) Electron	parti-penguin	6/8/18 at 9:59am	...
2b002c001847373239323130	(E) Electron	parti-bison	5/20/18 at 6:50pm	...
2e0024000a47373236303037	(E) Electron	parti-monkey	5/19/18 at 11:10am	...
1e0047000751373239383834	(E) Electron	parti-egret	5/19/18 at 11:09am	...

1 2 3 »



Particle

Devices

Number of devices: 27

ID	Type	Name	Last Handshake
3b001e000247363339343638	(P) Photon	trash-panda	7/19/18 at 8:57am
190028000247343138333038	(P) Photon	office_sensors	7/19/18 at 8:24am
638	(P) Photon	rotator	7/18/18 at 8:29pm
638	(P) Photon	wise-pirate	7/18/18 at 4:45pm
638	(C) Core	carrot_pants	7/17/18 at 5:19pm
1a0038001647373334363431	(P) Photon	avocado-toast	7/17/18 at 9:20am
2b002c001847373239323130	(E) Electron	parti-penguin	6/8/18 at 9:59am
2e0024000a47373236303037	(E) Electron	parti-bison	5/20/18 at 6:50pm
1e0047000751373239383834	(E) Electron	parti-monkey	5/19/18 at 11:10am
	(E) Electron	parti-egret	5/19/18 at 11:09am

1 2 3 »

Their type and name

Particle

Devices

Number of devices: 27

ID	Type	Name	Last Handshake	...
3b001e000247363339343638	(P) Photon	trash-panda	7/19/18 at 8:57am	...
190028000247343138333038	(P) Photon	office_sensors	7/19/18 at 8:24am	...
1a0038001647373334363431	(P) Photon	rotator	7/18/18 at 8:29pm	...
2b002c001847373239323130	(P) Photon	wise-pirate	7/18/18 at 4:45pm	...
2e0024000a47373236303037	(C) Core	carrot_pants	7/17/18 at 5:19pm	...
1e0047000751373239383834	(P) Photon	avocado-toast	7/17/18 at 9:20am	...
	(E) Electron	parti-penguin	6/8/18 at 9:59am	...
	(E) Electron	parti-bison	5/20/18 at 6:50pm	...
	(E) Electron	parti-monkey	5/19/18 at 11:10am	...
	(E) Electron	parti-egret	5/19/18 at 11:09am	...

And the last time they appeared online

1 2 3 »

# REAL-TIME EVENT LOGS

**Devices > View Device**

**Particle**

ID: 3b001e000247363339343638 Name: trash-panda

Device OS: 0.8.0-rc.8 Type: P Photon

Serial Number: PHHMAB819ZY6QXD Last Handshake: Jul 19th 2018, 4:56 pm

**PING** **EDIT**

**Notes**  
Click the edit button to keep notes on this device, like 'Deployed to customer site'.

**EVENT LOGS** **DIAGNOSTICS NEW**

spark/flash/status device went offline device came online spark/device/last\_reset  
spark/device/diagnostics/update spark/device/app-hash

04:49:00 PM 04:50:15 PM 04:51:30 PM 04:52:45 PM 04:54:00 PM 04:55:15 PM 04:57:35 PM

**PAUSE** **SEE IN TERMINAL** **PUBLISH EVENT**

EVENT NAME	DATA	PUBLISHED AT
spark/device/diagnostics/update	{"device":{"system":	July 19th at 4:56:39 pm
spark/device/last_reset	power_down	July 19th at 4:56:39 pm
device came online		July 19th at 4:56:39 pm

**DEVICE VITALS NEW**

Jul 19th, 2018, 04:22PM

- Strong Wi-Fi signal
- 0 disconnect events
- 146ms round-trip time
- 0 rate-limited publishes
- 35kB of 81kB RAM used

[Download History](#) | [Run diagnostics](#)

**FUNCTIONS**

f updateFName = 1

Brandon **CALL**

f updateLName = 1

# SIM MANAGEMENT

#PartiBadge | Electron | 7461

Docs | Contact Sales | Support | bsatrom@gmail.com

## SIM Cards

MB

Aug 7 Sep 7

Data Usage i  
**6.24 MB**  
used since Aug 8th

Active SIMs i  
**12**  
in product fleet

+ IMPORT SIM CARDS

SET DATA LIMIT

Status	ICCID	Device ID	Device Name	Data limit
Active	...5585	none	none	10 MBs
Active	...5113	52004...	parti-llama	5 MBs
Active	...2212	3c004...	parti-zebra	5 MBs
Active	...6163	2b002...	parti-bison	5 MBs
Active	...3921	28003...	parti-lemur	5 MBs
Active	...8186	25002...	parti-alpaca	5 MBs
Active	...1156	20003...	parti-nerfherder	5 MBs
Active	...1313	1e004...	parti-egret	5 MBs
Active	...9135	1e004...	parti-parrot	5 MBs

Filter by ICCID i ICCID ▼

DATA USAGE

10MB  
7MB  
Aug 8 Sep 8

**6.24 MB**  
used since Aug 8th

**3MB included/mo.**  
**\$0.40+ addnl. MBs**

# SIM MANAGEMENT

#PartiBadge | Electron | 7461

Docs | Contact Sales | Support | bsatrom@gmail.com

## SIM Cards

Data Usage i  
**6.24 MB**  
used since Aug 8th

Active SIMs i  
**12**  
in product fleet

+ IMPORT SIM CARDS

SET DATA LIMIT

Filter by ICCID ICCID ▾

Docs

Status	ICCID	Device ID	Device Name	Data limit
Active	...5585	none	none	10 MBs
Active	...5113	52004...	parti-llama	10 MBs
Active	...2212	3c004...	parti-zebra	5 MBs
Active	...6163	2b002...	parti-bison	5 MBs
Active	...3921	28003...	parti-lemur	5 MBs
Active	...8186	25002...	parti-alpaca	5 MBs
Active	...1156	20003...	parti-nerfherder	5 MBs
Active	...1313	1e004...	parti-egret	5 MBs
Active	...9135	1e004...	parti-parrot	5 MBs

**DATA USAGE**  
**6.24 MB**  
used since Aug 8th

**3MB included/  
\$0.40+ addnl. usage**

**Set SIM Data Limit**

**Service will be paused** once your SIM has reached the data limit below:

Data Limit  
10MB

**SET DATA LIMIT**

*It may take up to one hour to pause your SIM once it has reached its data limit.  
Overage costs in that timeframe may apply.*

# REMOTE DIAGNOSTICS

EVENT LOGS **DIAGNOSTICS NEW**

 RUN TESTS

 **EVERYTHING LOOKS GOOD!**

All diagnostic tests have passed. This device is healthy.

 **Device Vitals**

 **Healthy**

▼

-  **Strong Wi-Fi signal** ⓘ
-  **0 disconnect events** ⓘ
-  **57ms round-trip time** ⓘ
-  **0 rate-limited publishes** ⓘ
-  **33kB of 81kB RAM used** ⓘ

 **Device Cloud**

 **Healthy**

▼

-  **API**
-  **Device Service**
-  **Webhooks**

# REMOTE DIAGNOSTICS

EVENT LOGS **DIAGNOSTICS NEW**

 RUN TESTS

 **EVERYTHING LOOKS GOOD!**

All diagnostic tests have passed. This device is healthy.

 **Device Vitals**

 **Healthy**

▼

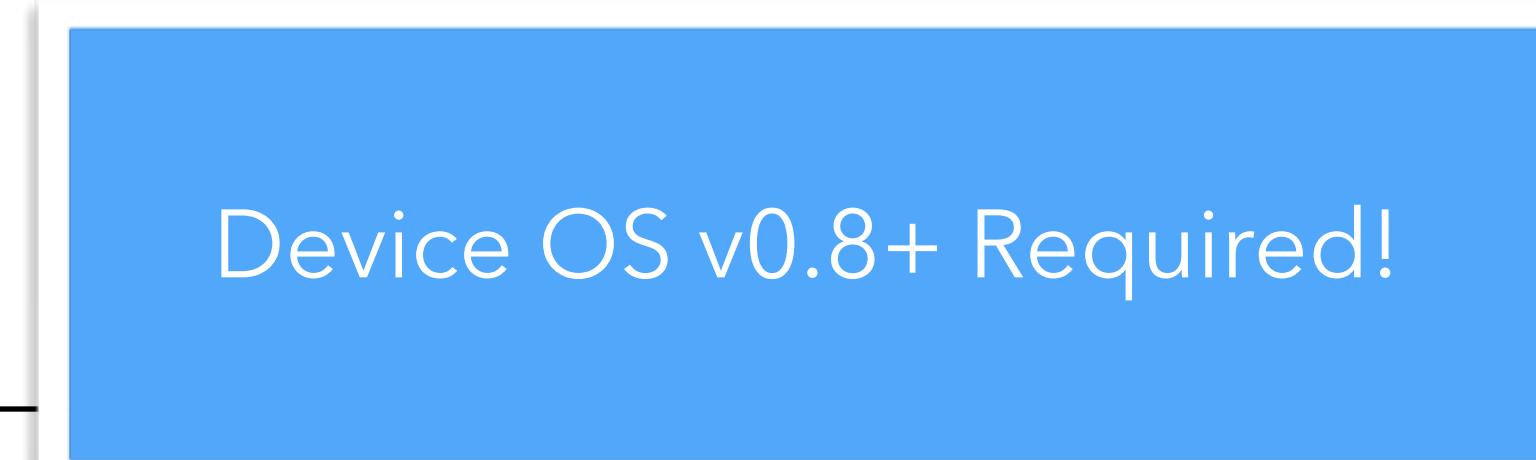
-  **Strong Wi-Fi signal** ⓘ
-  **0 disconnect events** ⓘ
-  **57ms round-trip time** ⓘ
-  **0 rate-limited publishes** ⓘ
-  **33kB of 81kB RAM used** ⓘ

 **Device Cloud**

 **Healthy**

▼

-  **API**
-  **Device Service**
-  **Webhooks**

 Device OS v0.8+ Required!

# VIEWING CLOUD VARIABLES AND CALLING CLOUD FUNCTIONS

## FUNCTIONS

**f updateFName**

Brandon CALL

**f updateLName**

Satrom CALL

**f checkTemp = 1** Argument CALL

## VARIABLES

✓ wearerFName (string) = **Brandon** GET

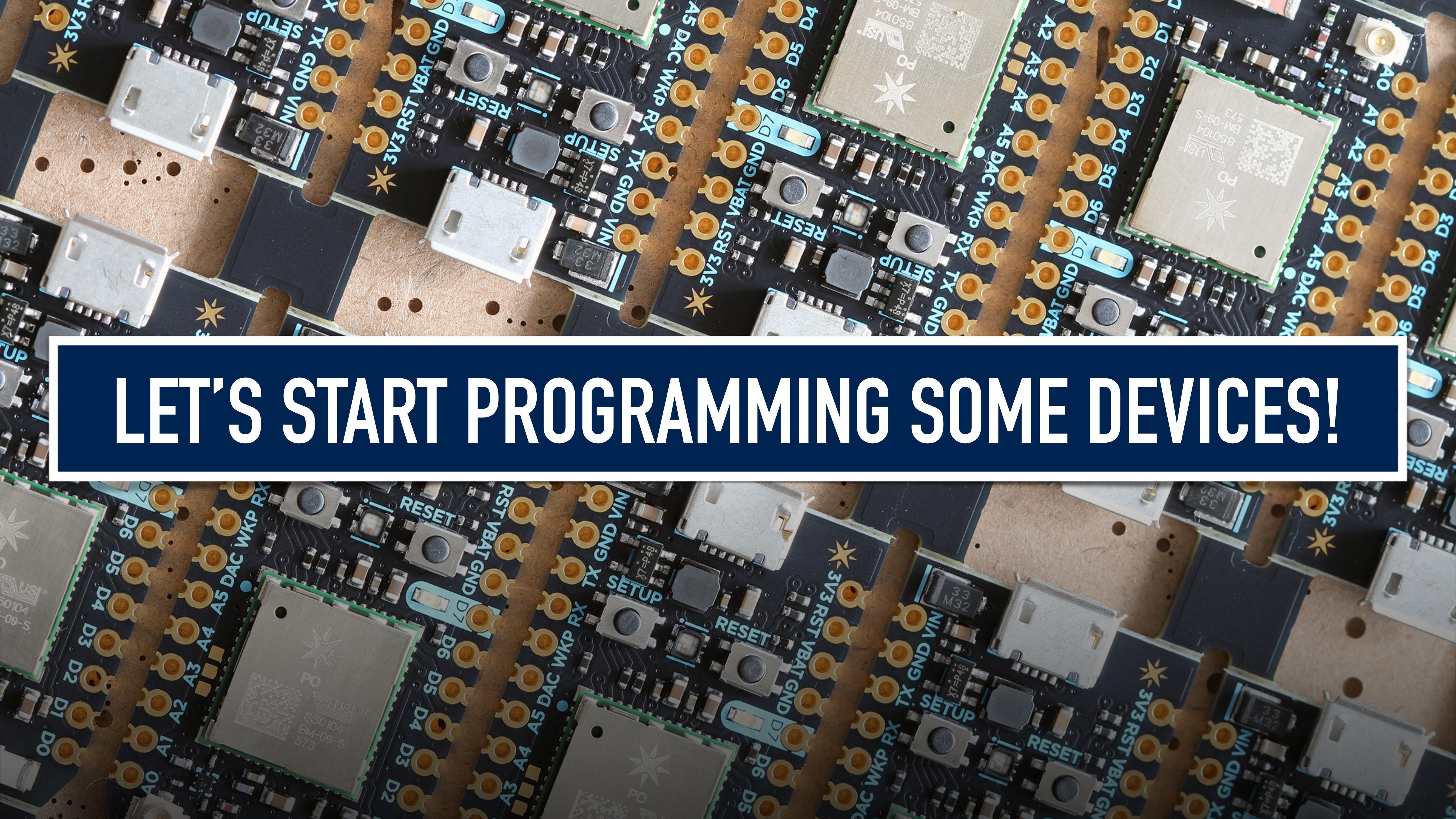
✓ wearerLName (string) = **Satrom** GET

✓ currentTemp (int32) = **76** GET

✓ currentHu (int32) = **29** GET

# THE PARTICLE CONSOLE

DEMO



# LET'S START PROGRAMMING SOME DEVICES!